

2024 年（第 17 届）
中国大学生计算机设计大赛

人工智能实践赛作品报告

作品编号：2024019207

作品名称：基于深度学习算法的非侵入式电力负载分类与预测系统

填写日期：2024 年 04 月 25 日

参赛人员：杜宇、姜川、李晓语、李庆隆、张一雯

填写说明：

1、本文档适用于人工智能挑战赛预选赛；
2、尽管预选赛仅完成部分工作，但是本文档需要针对决赛做出方案设计；
3、正文、标题格式已经在本文中设定，请勿修改；标题#的快捷键为“Ctrl+#”，正文快捷键为“Ctrl + 0”；
4、本文档应结构清晰，突出重点，适当配合图表，描述准确，不易冗长拖沓；
5、提交文档时，以 PDF 格式提交；
6、本文档内容是正式参赛内容的组成部分，务必真实填写。如不属实，将导致奖项等级降低甚至终止本作品参加比赛。

目 录

第1章 作品概述

1.1 作品背景

以数字经济助力节能减排，以高新技术助力降本增效，是各行各业千家万户的大势所趋。然而，我国目前的电力资源的不合理利用的现象仍然非常严重，有待我们去解决。中国节能产品认证中心的调查发现，一个普通城市家庭平均每天的家电待机时都会造成一定能耗。全国 4 亿台彩电一年的待机耗电量就高达 29.2 亿度，相当于大亚湾核电站全年 1/3 的发电量。因此，以电力监测的方法发掘、控制电力资源浪费的必要性日益凸显，而非侵入式技术，则又是前者的重中之重。我们小组研发的这款基于非侵入式技术的电力负载分类分解与功率预测系统便是利用人工智能技术为家庭和小型企业减少用电开销、推动节约电力资源的真实写照。

1.2 作品简介

基于上述背景，我们设计出了一种面向家庭用户和小规模企业的电力负载分类与预测系统。其中，“分类”功能可实时告知用户各电器设备的运作状态；“预测”功能则能预估用户未来短时间内的电能消耗。无论是普通家庭还是企业工厂，都亟需这样的一套智能算法和系统来监控电器的实时运行状态，识别不必要的电力消耗。这不仅有助于节约能源，还能降低用电成本。预测短期电能消耗的功能还能使企业更有效地规划电力使用，特别是在实行阶梯电价机制的情况下。

比如家庭用户登录我们的系统后，可以查看住宅中实时耗电功率的波形曲线图、近 50 分钟房屋内消耗的总电能，以及模型判断正在运行的电器（如：“热水器”，“电脑”，或者是多种电器的组合等等）。用户还可以查看 AI 模型推测后 20 分钟房屋内的电能消耗情况（预测的功率曲线）。

我们研发的系统一个重要的特点是“非入侵”：在不监测具体电器设备的情况下，通过分析用电单位整个电力系统的总用电数据来分析各个电器设备用电情况并预测功耗。其背后的 AI 算法，采用了级联模型结构，使用深度神经网络模

型+PCA 主成分分析算法进行分类，并仿照基于 Transformer 的生成式语言模型的工作原理以及 FFT 进行短时功率数值预测。

应用部署方面，系统采用“应用分布，算力集中”云端协同的架构，包含边缘设备（电能表）、分布式的后端和数据库、位于超算中心的分类与预测 AI 模型，以及位于移动设备的微信小程序客户端。微信小程序更易普及，系统的监测和预测功能对用户起到提醒作用，这样每个用户都能够注意电器能耗，减少不必要的浪费，从而助力节能减排。这样的架构设计科学合理，可满足供电用电单位需求，耦合度低，且很容易拓展系统规模，具有良好的发展前景。目前我们的算法肯定还有提升空间，相信通过未来的努力和时代的推动，我们的算法会创造更多的价值。

第2章 问题分析

2.1 问题来源

前文已讲到节能减排与降低用电成本的重要性。但要解决这个问题，不可避免的就是电力的监测。但随着智能电网的迅速发展，大数据已成为重要因素，面对海量数据需求，电力的侵入式(直接面向电器设备)监测方法已经很难满足需求。传统方法监测设备硬件繁杂、投入过多、管理复杂，甚至部分数据无法监测识别的问题预示着这些方法的淘汰。于是乎，我们采用了简洁高效的非侵入式方法来解决上述问题。

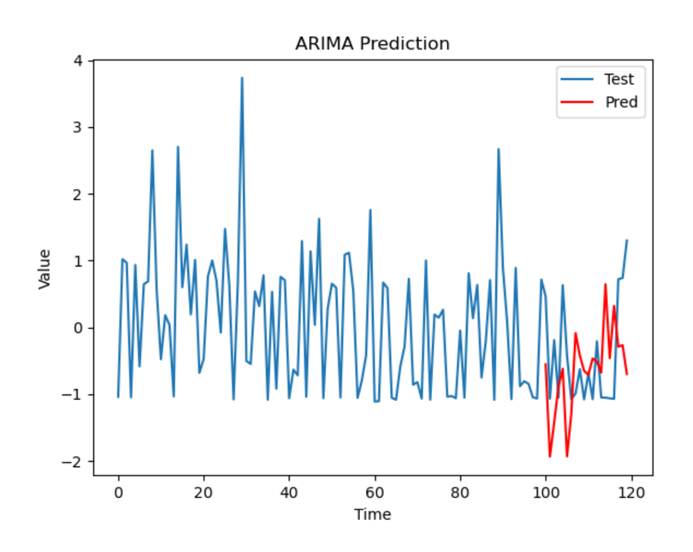
2.2 现有解决方案

面对日益增大的电力系统规模，现仍存在的侵入式电力负载监测产品无力解决，只能日渐淘汰。

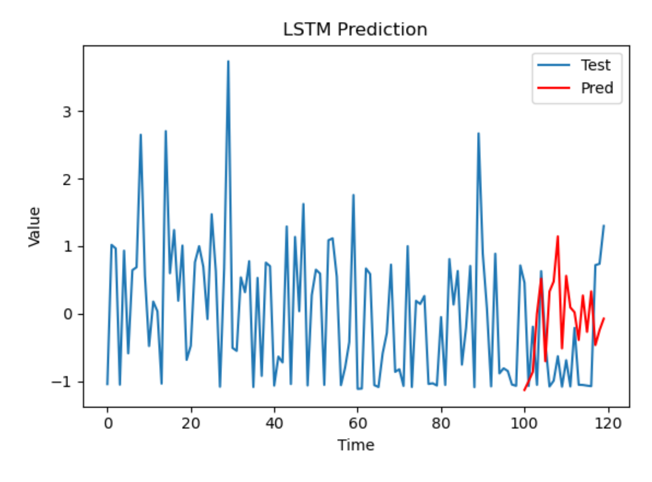
目前市面上的相关产品主要使用的非侵入式分类分解算法有基于传统机器学习的决策树、随机森林算法、SVM（支持向量机）等，异或是利用集成学习算法对其进行集成。负载预测算法主要有回归分析算法、ARIMA 算法和基于深度学习的 LSTM 算法等。以上算法的预测准确率都不理想，并且不具有电力负载场景

的针对性—它们只是解决了普通的时间序列预测或分类问题，而没有考虑电功率数据特有的一些性质和处理方法。

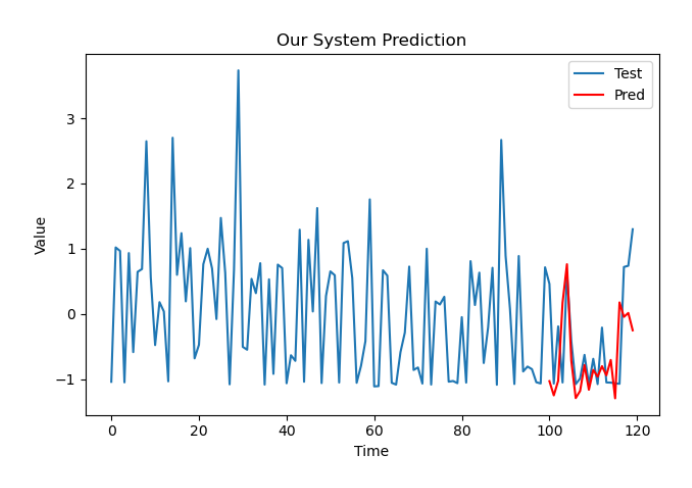
以下是对于完全相同训练集和测试集下的，基于 ARIMA 算法、LSTM 算法以及我们的系统的预测功率曲线图：



【图 1】基于 ARIMA 算法的预测功率曲线



【图 2】基于 LSTM 神经网络的预测功率曲线

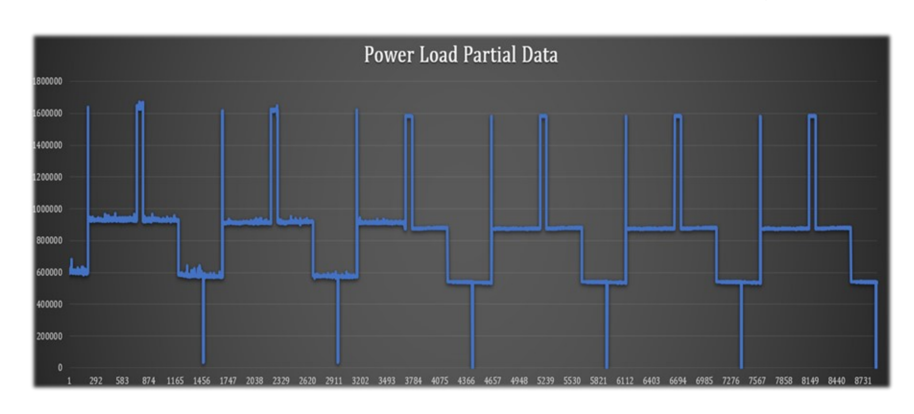


【图 3】我们系统的算法预测的功率曲线

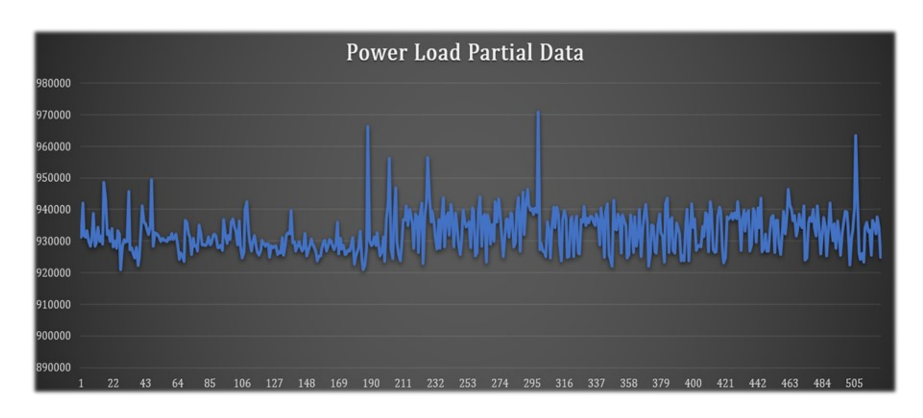
2.3 本作品要解决的痛点问题

痛点 1：短时电力负载功率数值预测的难题

首先给出同一时刻下边缘设备电能表在现实中采集到的某典型家庭的电力负荷总数据，图 4 的时间粒度为 1 分钟，时间跨度为 8700 分钟。图 5 的时间粒度为 1 分钟，时间跨度为 500 分钟。（单位：*10^-3 瓦特）



【图 4】采集到的长时功率曲线 1



【图 5】采集到的短时功率曲线 2

我们系统的预测功能是指短时的电力功率时间序列预测。然而电力负载时序数据具有的特点是：我们可以很直观的感受到，当用电器种类与状态不变时，宏观上非常具有规律性，极易预测；而当时间跨度缩小时，序列在微观上就好像完全没有了规律，基于短时间检测值的功率预测将变得非常困难。究其原因，不外乎短时功耗受外部不确定性因素影响太大。在 2.2 节中我们提及的 LSTM 算法和 ARIMA 算法在长时间跨度下的功率预测任务上表现效果尚可，但对于短时间跨度的功率预测问题却表现得非常差。我们能做的，就是从万变中找不变，设计更加科学的人工智能与数学的算法找出数据中的“主要矛盾”。

痛点 2：大规模用户并发请求造成模型无法及时推理的问题

我们考虑到了若按照传统的“客户端的请求触发模型推理”的工作模式，如果有短时间内突然有大规模的用户请求，服务器集群性能不变的情况下，会出现算力请求拥堵的情况，具体表现在用户打开客户端，会等待很长时间才能看到模型计算出的分类与预测值。我们的解决方法是，设计一套不同于传统模式的“云端协同”系统架构，确保 AI 模型端可以不间断运行计算，用户可以及时查询电力负载情况。具体内容参见第 3 章技术实现部分。

2.4 解决问题的思路

我们的系统功能上是面向家庭用户和小规模企业的电力负载分类与预测。其中，“分类”功能可实时告知用户各电器设备的运作状态（“分解”也通过分类的功能去实现：我们将多个电器的组合也看作一个类别，当算法将其识别后，即为完成了分解）；“预测”功能则能预估用户未来短时间内的电能消耗。比如家庭用户登录我们的系统后，可以查看住宅中实时耗电功率的波形曲线图、近 50 分钟房屋内消耗的总电能，以及模型判断正在运行的电器(如：“热水器”，“电脑”，或者是多种电器的组合等等）。用户还可以查看 AI 模型推测后 20 分钟房屋内的电能消耗情况（预测的功率曲线）。

我们训练模型并评估算法时使用的数据集是电能表实际采集到的普通家庭用户电力负载时序数据，数据来源是济南相关电力企业在济南市约十个小区中随机抽取的数百家住户。企业有告知用户在特定时间段内按照要求只开启指定电器，保证了数据的准确性。数据包含两种类型：21 类两两电器组合运作的数据以及 7 种单电器单独运作的数据。数据包含 USB 风扇、电吹风、电磁炉、白炽灯、电冰

箱、电脑、热水壶等电器及其组合。数据粒度为 1 分钟，数据规模为每个类别大约 20 万分钟。由于数据集暂不可公开，故我们也可以利用开源电力负载数据集，同样使用我们的算法进行模型训练、部署、展示，我们的算法适于任何电力负载设备，并不针对特定的数据集。

可代替的开源电力负载数据集有：

BLUED http://portoalegre.andrew.cmu.edu:88/BLUED/

Dataport by PecanStreet https://www.pecanstreet.org/dataport/about/

EMBED http://embed-dataset.org/

UK-DALE https://ukerc.rl.ac.uk/DC/cgi-bin/edc_search.pl?GoButton=Detail&WantComp=41

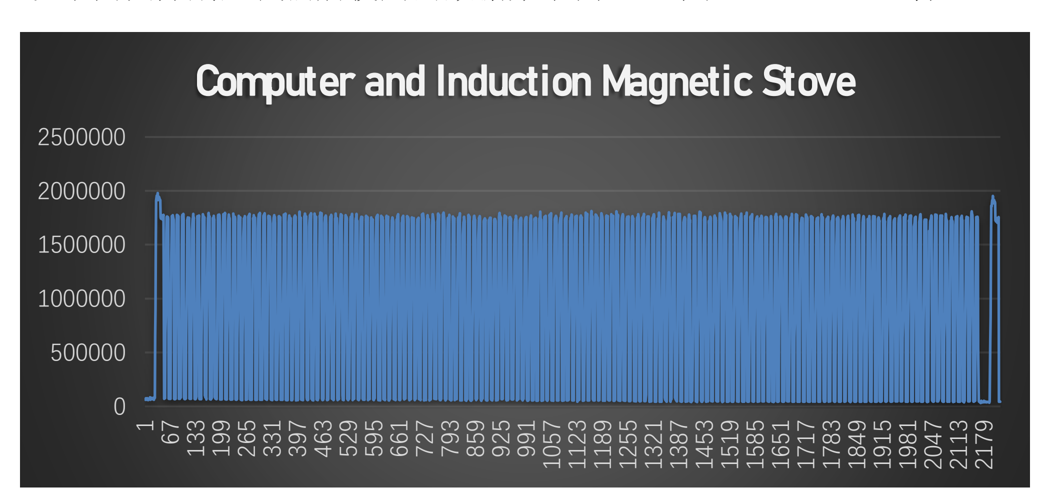
阿里天池电力 https://tianchi.aliyun.com/competition/entrance/231602/information

Smart* http://traces.cs.umass.edu/index.php/Smart/Smart

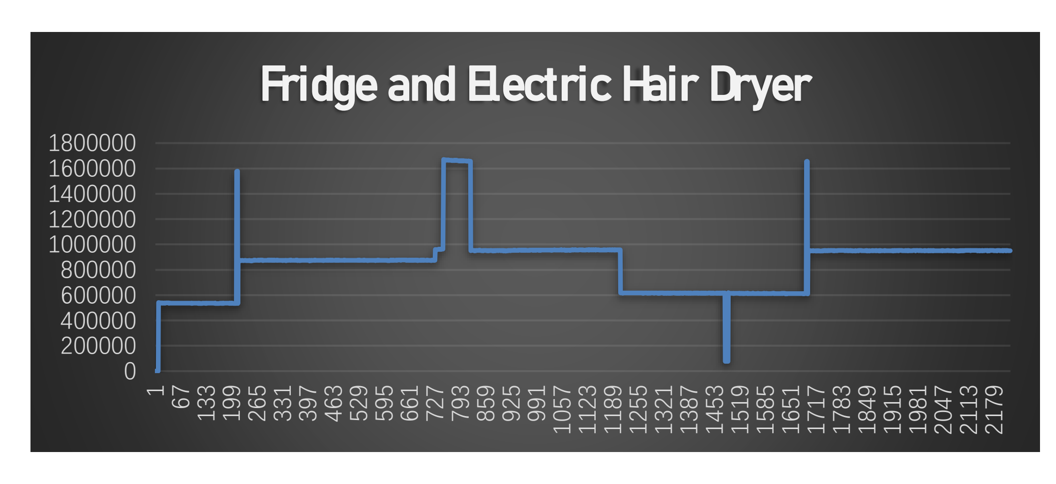
iAWE https://link.zhihu.com/?target=http%3A//energy.iiitd.edu.in%3A5000/

数据特点方面在前文也有说明，其在长时间跨度上可以突显出一定规律，但是在短时间内很难拟合出数据的规律和特征。

以下两图分别是我们所使用的数据集示例：（单位：*10^-3 瓦特）



【图 6】电脑与电磁炉数据示例



【图 7】冰箱与电吹风数据示例

系统设计大致思路：

AI 模型：采用级联模型结构，将任务分为分类分解与预测两个级联的模块。分类采用深度全连接神经网络模型去处理，根据时间窗的特征，实现分类；分解任务通过分类任务去实现；预测模型受生成式语言模型的启发而设计，采用了以 Transformer 结构和为核心的深度神经网络两个模型通过电器类型标签相关联。具体内容参见第 3 章技术实现。

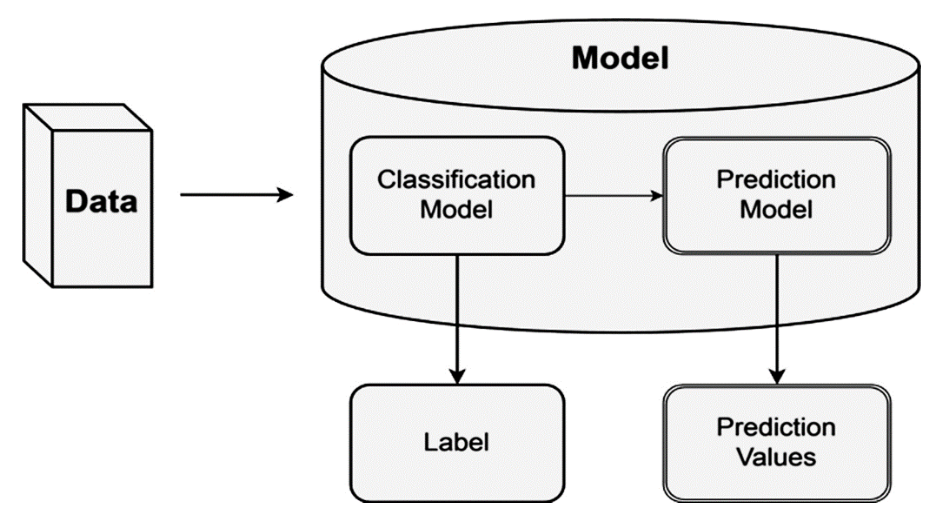
系统架构：采用适于现代化超算服务平台的“应用分布，算力集中”云端协同架构。包含边缘设备（电能表）、分布式的后端和数据库、位于超算中心的分类与预测 AI 模型，以及位于移动设备的微信小程序客户端。这样的设计可以在一定程度上解决前文所提及的大规模用户并发请求造成模型无法及时推理的问题。

前端与后端设计：我们主要使用了微信开发者工具进行前端开发，波形绘制使用了 ECharts 库中提供的折线图组件。前台轮询查询数据。为提高性能，后端程序使用 C++17 构建 Server，其基于单 Reactor 多线程架构。服务器系统核心由日志记录、线程池管理、IO 多路复用、HTTP 处理、缓冲区管理和阻塞队列等模块组成。HTTP 请求报文通过分散读方式读入，并利用有限状态机与正则表达式进行高效解析。

第3章 技术方案

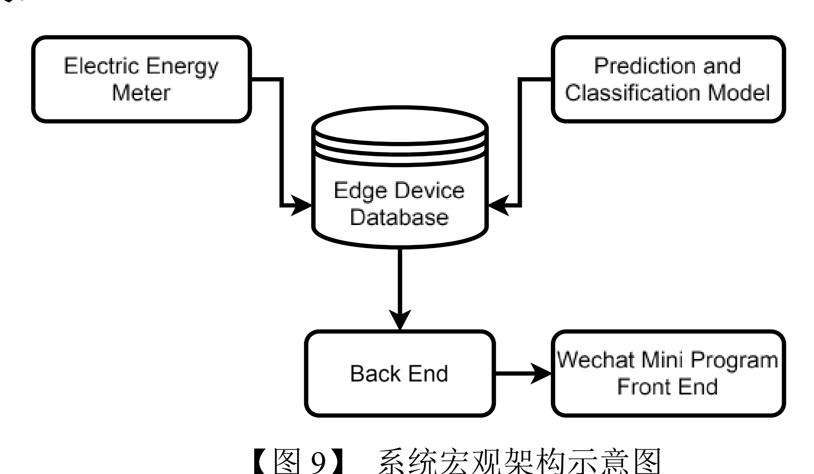
3.1 技术总体概述

模型设计概述：我们的系统采用了级联模型架构，模型包括分类分解模块和数值预测模块。输入数据首先通过分类模块进行标签化处理，经过处理后，数据及其标签被输入至数值预测模块，以生成最终的功率预测值。通俗来讲即为，模型分为分类分解模型与数值预测模型两个部分。首先数据会被前者进行分类，然后数据以及通过分类得到的标签会再被送入数值预测模型中，输出预测的功率数值。在下文中，我会分别讲述两者的详细原理。模型的工作流程如图 8 所示。



【图 8】 分类与预测模型工作流

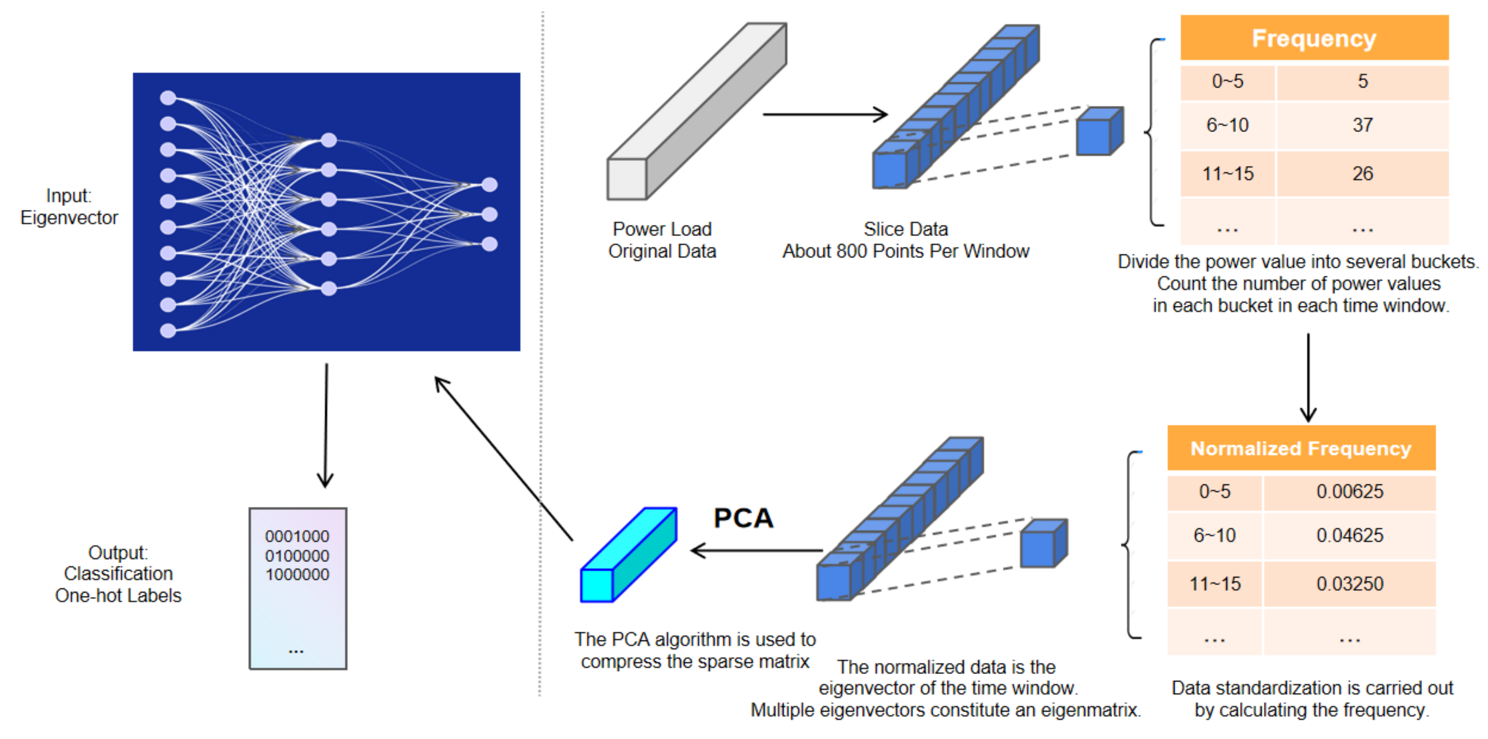
系统部署概述：项目的部署采用了适于现代化超算服务平台的“应用分布，算力集中”云端协同架构。其中电能表属于边缘设备，用于采集各用电单位的耗电数据，将采集结果写入数据库。数据库采用 SQLite 轻量级数据库。MQ 的设计则削减了来自分布式后端的请求洪峰，减轻了 GPU 集群的压力，提升系统整体性能。系统后端部署于小规模用电单位构成的集合中，比如：一个居民小区、一家公司等，后台为集合内的每个用电单位用户的微信小程序提供分类与预测数据转发服务。分类分解与预测 AI 模型部署于依托国家超算互联网强大算力的服务器集群。模型每隔一定的时间“主动”读取数据库中由边缘设备采集到的数据，并进行推理计算。完成后将计算结果写入对应数据库。后台处理客户端请求与向模型请求数据的两个过程是不相关的，两者互不影响，互不牵制。其响应客户端时只需读取数据库中的各项结果，不与 AI 模型端直接关联，降低了耦合度，且易于拓展。



【图 9】 系统宏观架构示意图

3.2 算法技术设计

3.2.1 电力负载分类分解算法设计



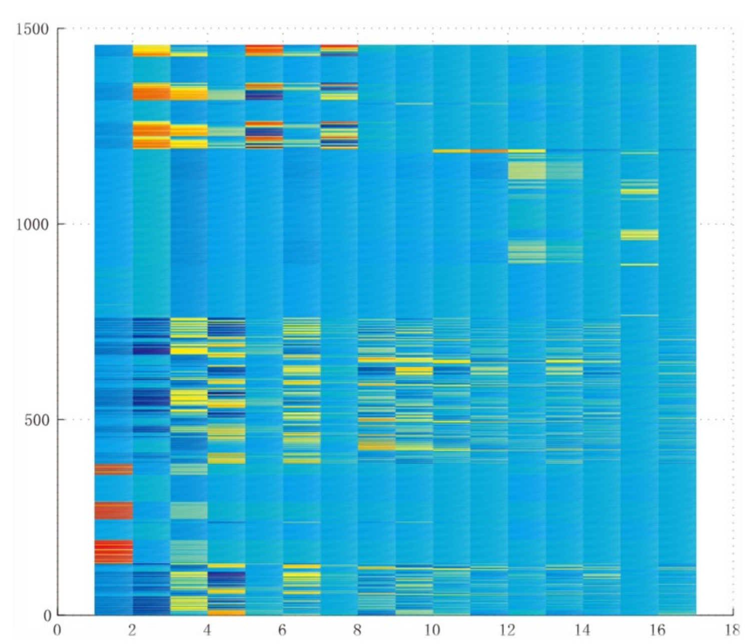
【图 10】 电力负载分类模型示意图

特征工程方面，首先对数据分帧，就是把每个标签对应的非常长的时序数组切分成长度相等的若干小段，每帧的长度(设为 L=800 分钟)作为参数；进一步我们对样本值的大小再作分组，比如（以下以 7 分类模型的训练举例）：功率大小(以下均以瓦特 W 为单位)从 0 开始，到 3300(设为 M=3300，作为参数)是最大值，中间每隔 5 个单位(设为 I=5，作为参数)分一组，则功率大小可以被分为 0 ~ 5，6 ~ 10，..... 3295 ~ 3300，共[M/I]=660 组。然后分别统计每一帧中的样本值分别落在这些组里的个数，这些数可以构成一个[M/I]维的向量 V。最后做归一化处理，将 V 除以帧长 L，可得该帧的特征向量 V/L=F。

上述计算过程中，我们采用了较短的窗长以及较多组的功率数值。根据迪利克雷原理，将会有很少的样本落入某个分组中，因而特征矩阵将会非常稀疏，不利于模型收敛，无法直接输入神经网络，需要经过主成分分析（PCA）算法的“压缩”处理。算法保留了 96%的方差，将 660 个维度的特征有效地压缩为了 17 维，

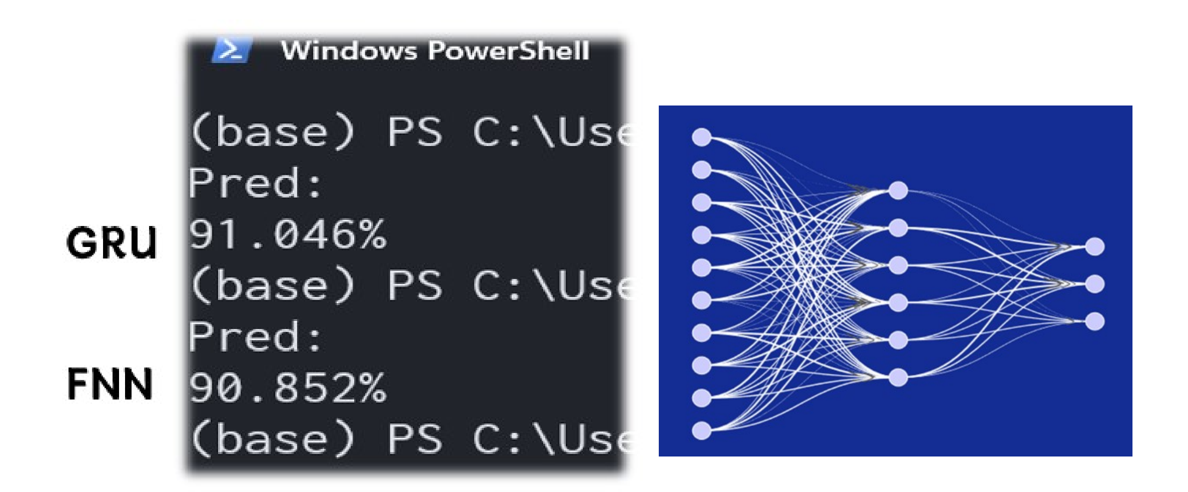
使特征矩阵成为了密集矩阵。

以上简而言之便是，先将输入的数据加窗处理，再将功率数值进行分组。接着，统计每个时间窗内落入各组的样本数量，得到频数分布。为了得到特征向量，我们对每个时间窗的频数进行归一化处理，即除以窗长。最后，使用 PCA 算法进行特征矩阵压缩。



【图 11】 经过 PCA 算法压缩后的特征矩阵热图(方差保留 96%)

模型设计方面，我们选用了全连接神经网络（FNN），输入该时间窗对应的特征向量，输出对应电器类别的独热编码。我们使用 3 层全连接神经网络模型(28, 14, 7) 进行数据分类任务（7 种单电器）；使用 3 层全连接神经网络模型(84, 42, 21) 进行功率标签分解任务（21 种电器两两组合）。选用模型时，我们对比了全连接网络与 GRU（循环门控单元）的性能。经过超参数的调整，两者分别可达到 91%和 90%的最高准确率，也就是说，两者的效果都比较好并且相差无几。但是经过实验与上线测试说明，GRU 由于要关注其他时间窗，其推理功耗明显大于全连接网络。另一方面，我们假定每个时间窗之间是互相时序不相关的，这样模型可以专注的处理每一个时间窗中的特征而做出判断，并将标签数据传给预测模型的嵌入层进一步处理，预测模型再利用时序特性进行预测，各司其职。综上两种原因而选择采用全连接网络。



【图 12】 GRU 与 FNN 效果对比　【图 13】深度神经网络示意图

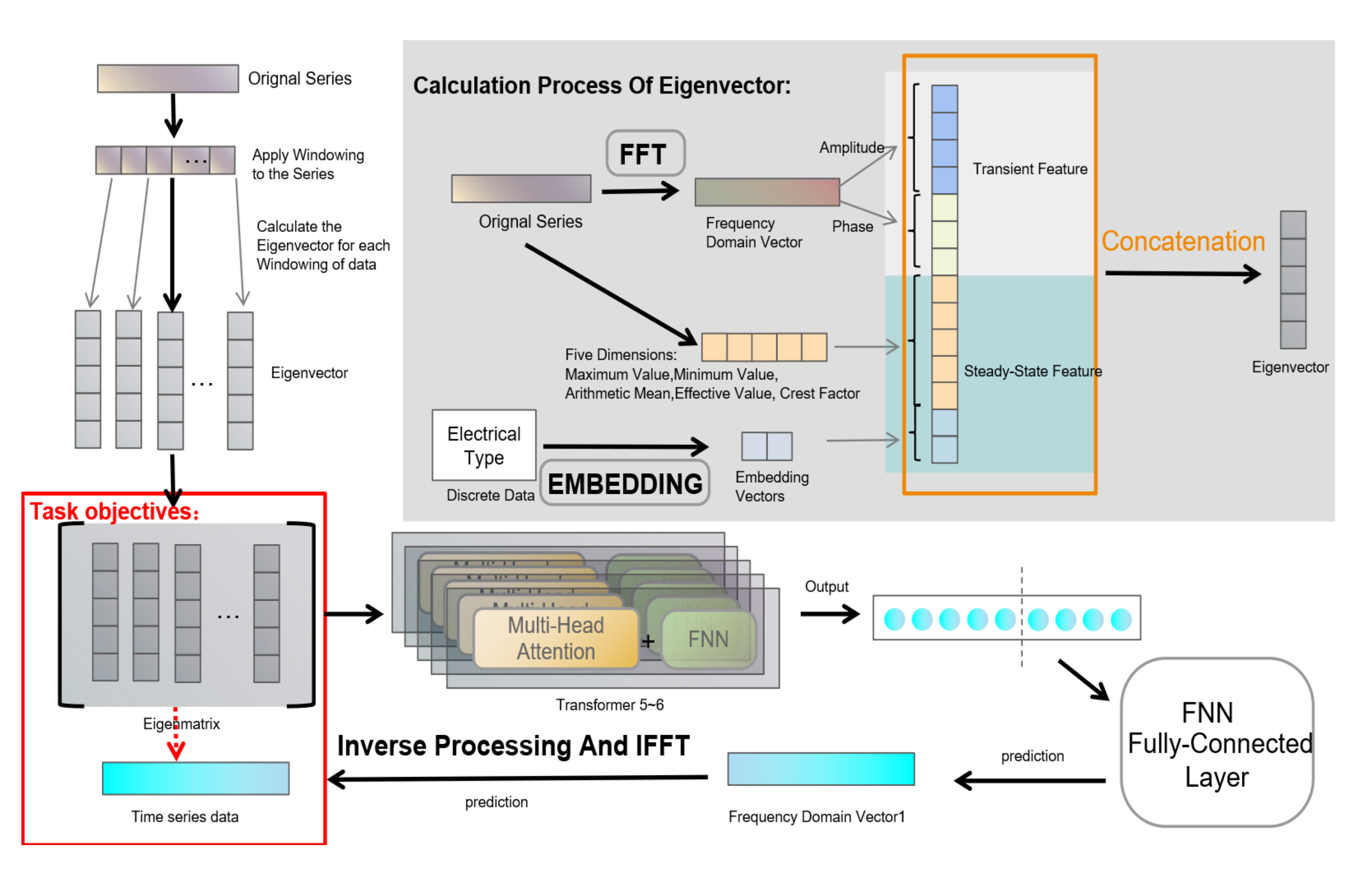
该网络结构的数学表达式可以表示为：

$$FNN(x) = \mathrm{ReLU}(\mathrm{ReLU}(xW_1 + b_1)W_2 + b_2)W_3 + b_3$$

$$Input\ Layer \in \boldsymbol{R}^{28} \qquad Hidden\ Layer \in \boldsymbol{R}^{14} \qquad Output\ Layer \in \boldsymbol{R}^{7}$$

3.2.2 电力负载预测算法设计

以下为我们发明的电力负载预测算法示意图。

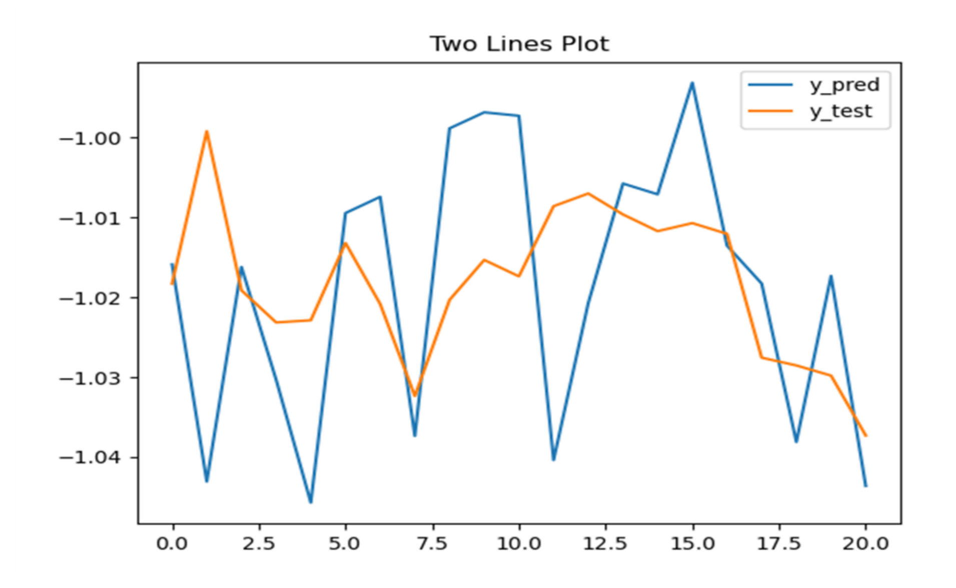


【图 14】电力负载功率预测算法示意图

对于电力功率时间序列预测的任务，我们设计了如下模型结构。在处理数据时，首先对数据进行分帧处理，然后在每个数据帧内进行分窗。每个时间窗都会计算出一个特征向量，多个时间窗的特征向量组成一个特征矩阵。这些特征向量由波动大的暂态特征和不易变的稳态特征两部分拼接而成。暂态特征的组成是通过快速傅里叶变换得到的各频率序数对应的振幅和相位（由于直流分量 DC 不包含任何有效信息，而 DC 所能反映的功率波形信号的采样均值可通过稳态特征体现，故我们将其去除）；而根据电学相关理论，稳态特征则由最大值、最小值、算术平均值、能量有效值和波峰系数等构成。此外，模型通过 Embedding 层将离散的电器种类映射为连续的嵌入向量，作为稳态特征的重要组成部分。受生成式语言模型的启发，特征矩阵被视为一句话的“词嵌入矩阵”，通过 5~6 层

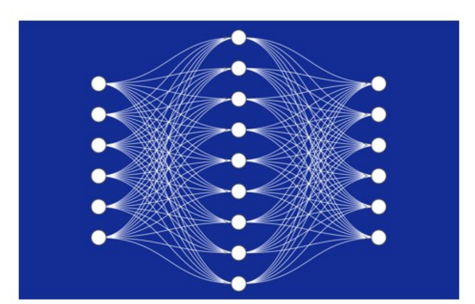
Transformer Block（最终选定了 6 层）和后续的注意力平均化处理，生成预测向量（相当于 LM 中生成的 token 对应的词嵌入向量）。最后，通过逆 FFT 将预测向量中的暂态特征（频域向量）转换为时间序列，从而完成功率时序预测。

但是经过训练试验发现，模型的损失值居高不下，并且它生成的频域向量补充 DC 后经过 IFFT 得到的时域波形的形状与测试集中的真实波形相差较大。(暂不考虑 DC 带来的波形偏移)



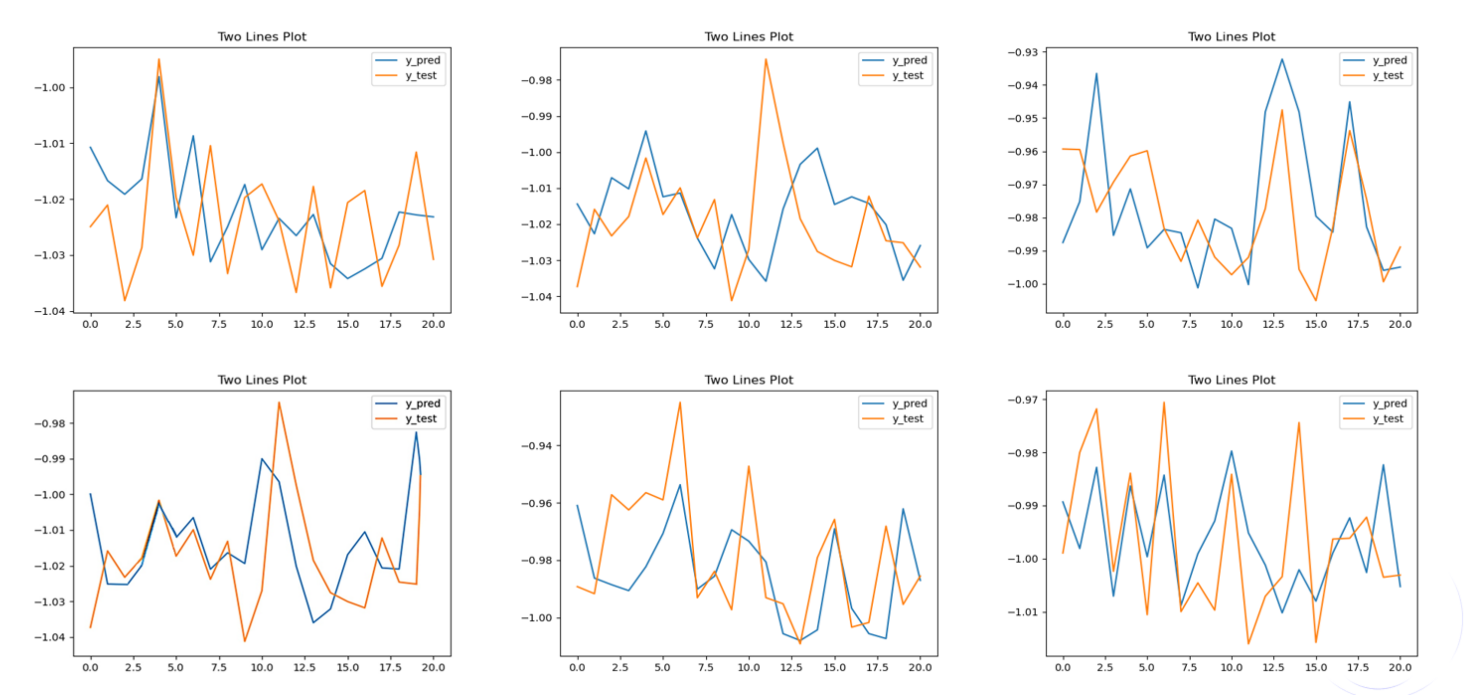
【图 15】改进前的功率预测波形对照

由 Mutihead 多头注意力输出的频域向量直接转换为时间序列，很显然这个效果并不是很好的，误差很大。我们考虑出现这个问题的原因在于，注意力机制只会根据每个时间步的特征的各个维度来计算出加权向量，它不会考虑一个时间步内部特征与特征各维度之间的关系，而全连接神经网络却可以很好的解决这个问题。因而我们的解决方案是在多头注意力层后面再添加一个全连接层，它的输入层与输出层均为每个时间窗的特征向量长度。



$Input\ Layer \in \boldsymbol{R}^{f_len}$ $Input\ Layer \in \boldsymbol{R}^{T\text{-}f_len}$

【图 16】在 Transformer 模型后添加全连接层的示意图



【图 17】经过改进后的模型在测试集上的对比

具体如下：

首先进行 FFT 计算，将时间窗内的时间序列信号数据转化为频域表示。转化后的信号长度与原始时序信号长度相等。其中第一个维度通常表示信号的 直流分量(DC)，由于其不包含信息，故我们将其去除。该计算过程可以表示为：

$$X[k]=\sum_{n=0}^{N-1}x[n]\mathrm{e}^{-j\frac{2\pi}{N}kn}$$

其中 N 表示信号长度，k 表示频率序数，x[n]为原始时序信号。事实上，上述表达式为离散傅里叶变换(DFT)的计算过程。在数学上，该表达式可以进一步优化，以便于计算机程序的执行。

$$X(f)=\int_{-\infty}^{+\infty}x(t)\mathrm{e}^{-j\cdot 2\pi ft}\,\mathrm{d}t$$

上述过程为快速傅里叶变换(FFT)。计算该函数时，首先进行信号长度的填充，使其长度为 2 的幂。然后使用分治策略和蝶形运算充分优化，此处不过多讨论。

经过计算得到的频域向量不能直接用作暂态特征向量，原因是未去除不包含有效信息的 DC 分量，并且通过 FFT 计算出的频域向量存在对称性的共轭冗余。对于 Numpy 的 FFT 计算函数，我们只需要得到除去了 DC 的一半向量即可获得该时间窗的所有信息。由于直流分量等于窗内数据的算术平均值，故以上操作等效于：（其中 X 表示 Numpy 计算出的祛除了共轭冗余的频域向量）

$$X_{\mathrm{RDC}} = \mathrm{X} - \frac{1}{N_X}\sum X$$

$$X_{\mathrm{T-feature}} = \mathrm{Concat}(Real(X_{RDC}), Imag(X_{RDC}))$$

特征向量即为暂态特征与稳态特征的合并：

$$X_{\mathrm{feature}} = \mathrm{Concat}\ (X_{\mathrm{T}-feature}, X_{S-feature})$$

稳态特征是指我们假定若电器的种类不变与运行工况没有出现大的变化时，该时间窗具有的特征，我们设计了窗内数据最大值、最小值、算术平均值、有效能量值以及波峰系数共 5 个固定维度以及可以根据数据规模更改的 n 维电器类型嵌入维度，共 n+5 个维度组成。以下是波峰系数计算方法和电器种类的映射示意：(其他从略)

$$Peak_Factor = \frac{max\ (X_i)}{\max\ (0, \sqrt{\frac{1}{ws}\sum {X_i}^2})}$$

$$Type_Vector = Embed_{\boldsymbol{Z}\to\boldsymbol{R}^n}\ (Type_i)$$

短时电能功率数据受当前时刻及其之前相近时间段的功率负荷变化的影响较大，而宏观上的数据变化趋势对微观时刻数据变化的影响微乎其微，因而我们决定使用自然语言处理领域中的注意力机制的算法，捕捉近距离若干时间窗的数

据中，哪些数据对接下来的“影响”比较大，“受影响”的权重是多少，进而预测新的数据。就好比自然语言处理中的生成式语言模型一样，通过提示词的词嵌入矩阵来预测下一个 token 的可能的词嵌入向量，从而生成下一个 token。可以表达为：

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

“token”的特征向量构成的矩阵 K 也可用其本身表示，再设 ws 为暂态特征频域向量长度（即为数据窗长度），稳态特征向量长度为 ss，故其注意力权重向量可以表示为：

$$Weight(X) = softmax\left(\frac{XX^T}{\sqrt{ws+ss}}\right)$$

其中激活函数 softmax 的表达式为:：

$$Softmax(z_i) = \frac{e^{Zi}}{\sum_{j=1}^{k} e^{z_j}}$$

算法预测下一时间窗的频率值的“依据”仍然为其自身，因此权重矩阵应与其作点乘，得到表示各频率分量以及其相位在各个时间窗内重要程度的加权后的向量。我们将这些向量在各个维度上的加权值求平均，得到预测的值。其中 fs 为数据帧的长度：

$$Y = Pred(X) = \frac{1}{fs}\sum \mathrm{Weight}(X)X$$

以上内容是一个自注意力层的预测输出，我们希望使用多个注意力层，结合它们的判断，做一个综合的输出。其中 Z 为多头注意力的头数。

$$Z = Mutihead(Y) = \mathrm{Concat}(head_1, \mathrm{head}_2, \ldots, \mathrm{head}_Z)W^o$$

$$head_i = \mathrm{Attention}(XW_i, XW_i, XW_i)$$

最终的输出值 Z 即为注意力层根据时序数据预测的特征向量值。

根据上文分析，我们还需将生成的 Z 值输入到一个全连接网络中，以让模型学习到各特征维度之间的关系。其最终的输出通过截取其中的暂态特征（频域向量）部分，再通过 IFFT 变换，即可实现时间序列预测。

$$Z_{\text{out}} = \text{FNN (Z)}$$

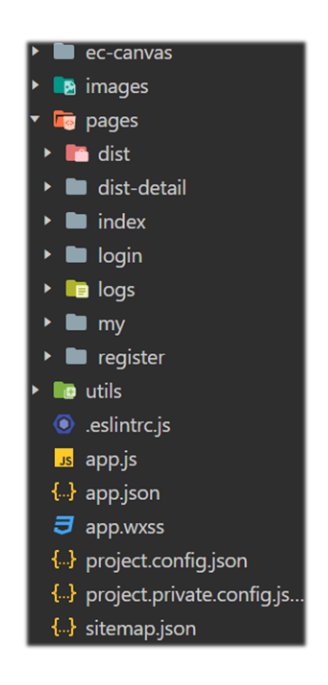
$$X_{\text{time}} = \text{IFFT (Freq}(Z_{out}))$$

3.3 系统其它部分的设计

3.3.1 前端与后端设计

前端实现：前端以微信小程序的形式实现，使用微信官方平台的微信开发者工具进行开发。图表展示采用了 ECharts 的折线图组件，每不到 1 分钟轮询请求一次图表数据（即给定长度数组）分别是实际和预测两种数据，用轮播图实现切换展示。用户信息添加到全局变量，在所有页面可共享，注销时清空。

后端概述：后端程序使用 C++17 标准构建的 Web Server，基于单 Reactor 多线程架构。系统核心由日志记录、线程池管理、IO 多路复用、HTTP 处理、缓冲区管理和阻塞队列等模块组成。HTTP 请求报文通过分散读方式读入，并利用有限状态机与正则表达式进行高效解析。



【图 18】前端代码结构

后端详细说明：总体框架采用的是单 Reactor 多线程模型，在主线程里面通过 I/O 多路复用监听多个文件描述符上的事件。主线程负责连接的建立和断开、把读写和逻辑处理函数加入线程池的任务队列，由线程池的子线程完成相应的读写操作，实现任务的高并发处理。在最底层实现了自动增长的 Buffer 缓冲区。对于 HTTP 请求报文，采用分散读进行读取，使用有限状态机和正则表达式进行解析；并通过集中写和内存映射的方式对响应报文进行传输。最后还加入了日志模块帮助工程项目开发和实现服务器的日常运行情况的记录。

首先是服务器的一个参数初始化操作。通过构造 WebServer 这个对象传递参数进行服务器相关参数的设定，主要参数有设置定时器超时时间、设置 Epoll 触发模式、设置日志系统的开启、日志路径以及异步开关、设置线程池的线程数量。然后是通过设定的参数对服务器的各个模块进行初始化。主要有日志、线程池、IO 复用、HTTP 对象、缓冲区、阻塞队列等模块。

日志模块的初始化是先建立一个日志文件，通过一个变量按照天数去划分不同的日志文件。线程池采用 RAII 手法，在构造时创建线程，析构时唤醒所有睡眠的线程使其退出循环。IO 复用是对 epoll 函数调用的一个封装。

HTTP 对象主要设置文件存放的相关路径。缓冲区和阻塞队列主要完成指定大小的参数设定。服务器各个模块初始化完成之后就是主线程里 I/O 复用监听事件的循环。监听事件有新连接到达、读事件和写事件和异常事件。根据不同的事件进行一个任务处理。

当新连接到达的时候，通过调用 accept 取出新连接（ET 模式下需要循环操作），将新连接的文件描述符用来初始化 HTTP 连接（套接字地址和文件描述符绑定到一个 HTTP 对象），完成绑定定时器的初始化，同时添加监听读事件，设置其文件描述符为非阻塞。

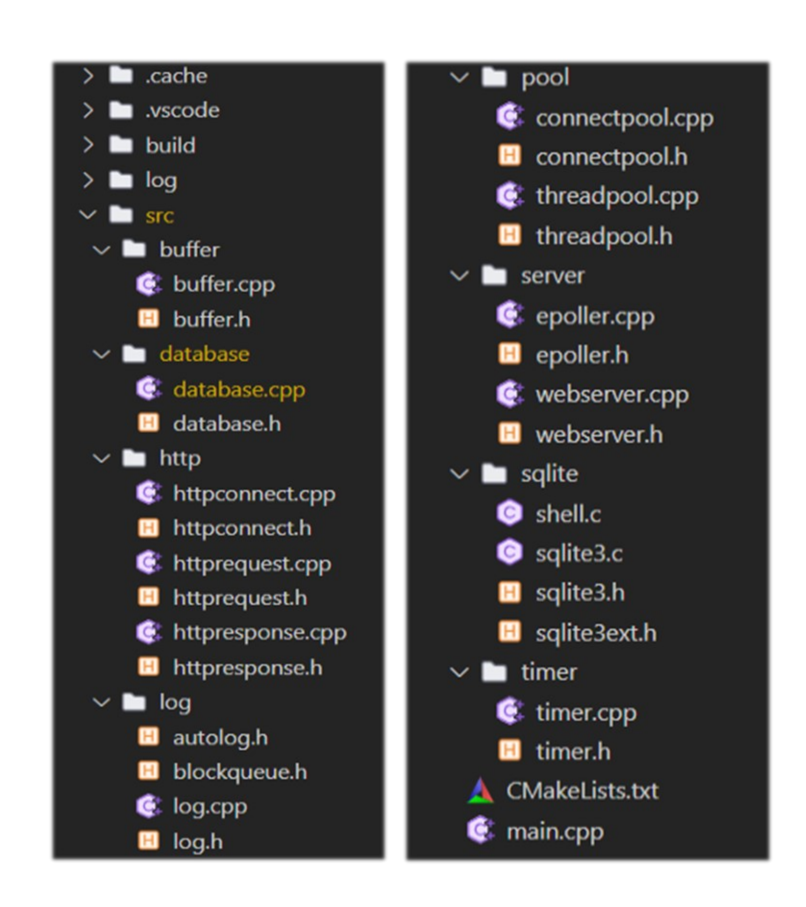
当有异常事件发生的时候，关闭该连接同时移除监听事件和定时器。当触发读事件的时候，调整定时器的定时时间，将读任务放入线程池的任务队列当中去。这个时候线程池对象里的多个线程，处于一个睡眠或者竞争任务并执行的过程，任务加入到任务队列当中去时会发送一个唤醒信号，如果有睡眠的线程则会被唤醒，进入循环里探测任务队列是否为空，取出任务并执行或者队列为空继续睡眠。线程执行读任务函数主要是完成一个非阻塞读取调用直到读完，将数据缓存在用户缓冲区中，接着执行一个消息解析的操作，根据 HTTP 解析是否成功的判断来决定重新注册写事件还是读事件。如果解析失败那么重新注册读事件等待下次读

取更多数据直到一个完整的 HTTP 请求。如果是解析成功的话就制作响应报文并且注册写事件，等待内核缓冲区可写触发事件时，将其写入内核缓冲区。

同时，基于心跳机制，每过一段时间会检测时间堆，去除掉长时间非活跃的连接。

这部分的重点是逻辑处理的过程，也就是 HTTP 解析和 HTTP 报文的制作。

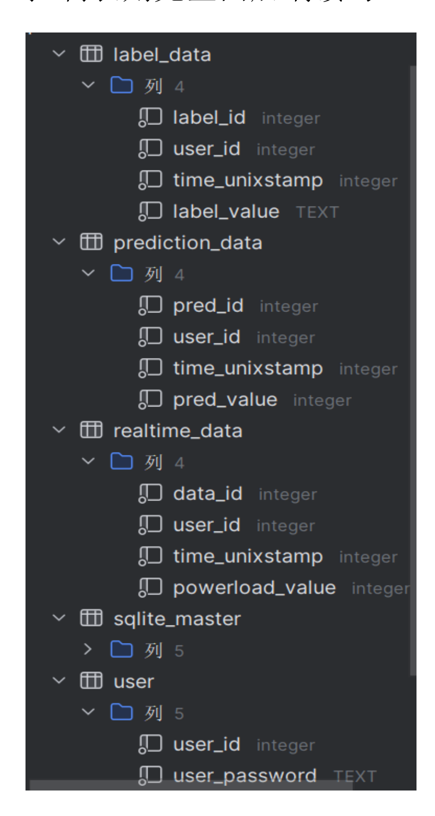
解析采用的是状态机和正则表达式，每次都读取以\r\n 结尾的一段字符串，通过状态机来判定获取的字符串是属于 HTTP 请求的哪一部分，再跳转到相应的函数进行解析，如果读取的字符串没有以\r\n 结尾则认为此次数据获取不完整，返回解析失败重新注册读事件。解析的时候判断是否是 POST 请求，如果是的话，还要解析 POST 的数据，里面包含了前端的请求信息，这部分是在数据体 BODY 部分，将 BODY 部分进行解析并存入到映射表里。制作响应报文时，根据从请求报文中解析出来的 POST 来判断前端想要请求的什么信息，从而进行 LOGIN, REGISTER, GETEDATA 等操作，并通过集中写将资源文件和响应报文分别发送回客户端。



【图 19】后端代码结构

3.3.2 数据库设计

数据库是整个系统的“枢纽”，连接边缘设备、后端以及位于算力中心的 AI 模型。故数据库的运行效率直接决定了整个系统的运行效率。数据库包含存储实时数据、预测数据、预测标签以及用户控制等表结构。其中实时数据由边缘设备写入，算力网络和后端读取。预测数据和预测标签由算力网络写入，后端读取。用户控制表则完全由后端读写。



【图 20】数据库表结构

第4章 系统实现

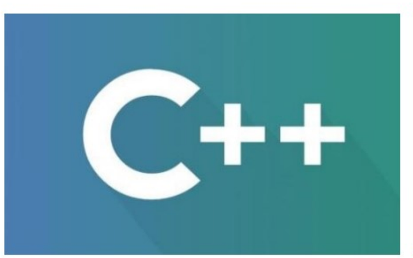
4.1 生产环境













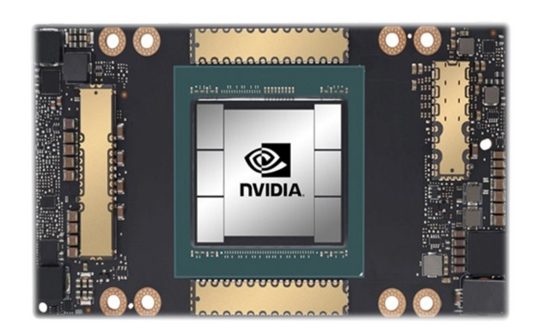
【图 21】本系统生产环境所涉及主要软件

4.2 模型训练与部署

4.2.1 硬件设备

模型部署设备：英伟达（NVIDIA） A100 GPU *1 服务器

模型训练本地设备：英特尔 第 2 代神经计算棒（Intel Neural Compute Stick 2）




```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.182.03   Driver Version: 470.182.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM...  Off  | 00000000:00:09.0 Off |                    0 |
| N/A   27C    P0    47W / 400W |      0MiB / 40536MiB |     22%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```




```
  - Batch:        1
  - Input layers:         Not specified, inherited from the model
  - Output layers:        Not specified, inherited from the model
  - Input shapes:         Not specified, inherited from the model
  - Mean values:  Not specified
  - Scale values:         Not specified
  - Scale factor:         Not specified
  - Precision of IR:      FP32
  - Enable fusing:        True
  - Enable grouped convolutions fusing:   True
  - Move mean values to preprocess section:       False
  - Reverse input channels:       True
nsorFlow specific parameters:
  - Input model in text protobuf format:  False
  - Path to model dump for TensorBoard:   None
  - List of shared libraries with TensorFlow custom layers implementation:        None
  - Update the configuration file with input/output node names:   None
  - Use configuration file used to generate the model with Object Detection API:  None
  - Use the config file:  None
del Optimizer version:

SUCCESS ] Generated IR version 10 model.
SUCCESS ] XML file: D:\Competition\Robot\Raspberry_Pi\Package\2IR\.\frozen_darknet_yolov4_model.xml
SUCCESS ] BIN file: D:\Competition\Robot\Raspberry_Pi\Package\2IR\.\frozen_darknet_yolov4_model.bin
SUCCESS ] Total execution time: 13.03 seconds.
's been a while, check for a new version of Intel(R) Distribution of OpenVINO(TM) toolkit here https://software.inte
```

【图 22】模型训练设备

4.2.2 训练过程

数据形式为 CSV 文件，数据总量为 250MB 左右，模型与数据的总体规模并不过于庞大，故我们没有专门为数据的导入而做过多的设计与优化。执行相应 Python 代码即可开始训练。在英特尔神经计算棒的加速下，提取分类模型特征需要 2 分钟左右的时间，训练只需 50 秒左右（21 分类）；训练数值预测模型需要 2 小时左右（21 类电器组合）。训练过程中不存在严重的梯度消失、梯度爆炸等不收敛的情况，训练中无需人为干预。

4.2.3 用户界面



【图 23】 用户界面

4.2.4 系统部署

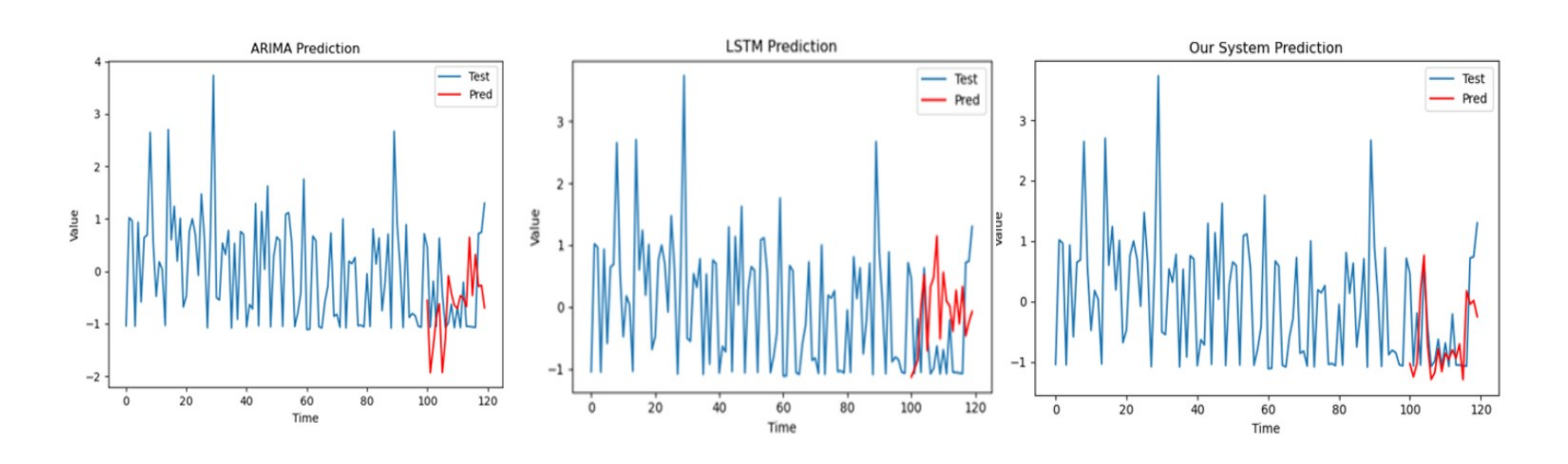
项目的部署采用了适于现代化超算服务平台的“应用分布，算力集中”云端协同架构。

其中电能表属于边缘设备，用于采集各用电单位的耗电数据，将采集结果写入数据库。数据库采用 SQLite 轻量级数据库。MQ 的设计则削减了来自分布式后端的请求洪峰，减轻了 GPU 集群的压力，提升系统整体性能。系统后端部署于小规模用电单位构成的集合中，比如：一个居民小区、一家公司等，后台为集合内的每个用电单位用户的微信小程序提供分类与预测数据转发服务。

分类分解与预测 AI 模型部署于依托国家超算互联网强大算力的服务器集群。模型每隔一定的时间“主动”读取数据库中由边缘设备采集到的数据，并进行推理计算。完成后将计算结果写入对应数据库。后台处理客户端请求与向模型请求数据的两个过程是不相关的，两者互不影响，互不牵制。其响应客户端时只需读取数据库中的各项结果，不与 AI 模型端直接关联，降低了耦合度，且易于拓展。

第5章 测试分析

前文中，我们已经将我们的模型与利用现行的算法训练的模型进行了对比，再对上文中的图形进行比对展示：



【图 24】三种算法对照图，最后一种是我们发明的预测算法的效果

另外，基于我们想法的构建的模型和系统架构的可行性也在现实中得到了验证。我们的一位指导教师已于近日将我们的作品的经过部分修改提供给了济南市某电力供应企业进行了小规模商用，规模约为两个小区指定的用户。目前系统运行良好，Beta 测试用户未反馈出现任何严重的问题。

由于测试用户相关资料以及企业对本系统具体修改的内容涉及企业秘密，在此恕不提供。

第6章 作品总结

6.1 作品特色与创新点

我们的作品系统，以自创的算法和架构设计解决了电力负载分类与预测应用场景中短时电力负载功率数值预测的难题和算力无法满足用户需求两大重点问题。系统有机地将非侵入式电力监测技术、深度学习算法与现代化软件系统架构相结合，实现了电力负载的智能分类与预测。采用 Transformer 算法等设计模型，紧跟学科前沿。其对于促进能源节约和电力管理现代化具有显著意义。

6.2 应用推广

应用将从微信小程序平台开始推广，从一部分小的群体开始，通过不断投入改善，使之面向更大的群体，如公司企业、学校等。以下是系统推广的几个关键点：

便捷易用的用户界面：采用微信小程序作为客户端，为用户提供了极为便捷的体验。用户只需打开微信，即可随时随地查看家中各电器设备的运行状态和未来电能消耗的预估情况。这种简单易用的设计将极大地促进系统的普及和应用。

宣传推广和用户教育：通过各种宣传推广活动，向用户普及节能减排的重要性和本系统的优势。可以利用社交媒体、线下活动等多种渠道，向用户展示系统的功能和效果，增强用户对系统的认知和信任。

合作共赢的商业模式：与能源供应商、家电生产厂商等相关企业进行合作，

共同推广和应用该系统。能源供应商可以通过系统提供的数据，优化能源供给和调度，提高能源利用效率；家电生产厂商可以通过系统提供的数据，优化产品设计和生产，提升产品的竞争力。这种合作共赢的商业模式将极大地推动系统的推广和应用。

总的来说，智能电力负载分类与预测系统的应用推广需要从用户体验、定制化服务、宣传推广和商业合作等多个方面入手，共同努力推动系统在家庭和社会中的广泛应用，为节能减排事业做出积极贡献。

6.1 作品展望

算法优化与性能提升：随着技术的不断进步，我们将致力于进一步优化系统的算法，提升其性能和准确度。通过引入更加先进的深度学习算法和数据处理技术，不断提高系统对电力负载的分类和预测能力，使其更加适用于不同类型的家庭和电力系统。

智能化能源管理：未来，我们将进一步探索智能电力负载分类与预测系统与智能能源管理系统的融合，实现对家庭能源的全面管理和优化。通过结合实时数据分析和人工智能算法，系统将能够更加智能地管理家庭能源使用，实现能源的高效利用和节约。

边缘计算与云端协同：随着边缘计算技术的不断成熟和普及，我们将探索边缘计算与云端协同的模式，实现对电力负载的实时监测和预测。通过在设备端进行数据处理和分析，减少对云端资源的依赖，提高系统的实时响应能力和稳定性。

生态系统的建设与开放合作：未来，我们将积极构建智能电力负载分类与预测系统的生态系统，与相关领域的企业、学术机构和社会组织开展开放合作，共同推动能源智能化和可持续发展。通过开放数据共享和技术交流，促进系统的创新和进步，为建设清洁、高效的能源未来贡献力量。

综上所述，智能电力负载分类与预测系统在算力方面的展望十分广阔，我们将不断探索和创新，以更加先进和高效的技术手段，为用户提供更优质的电力负载管理服务，助力能源智能化和可持续发展的实现。

参考文献

1. Ashish Vaswani, et al. "Attention Is All You Need" 2017.
(https://arxiv.org/pdf/1706.03762.pdf)